

Advanced Theming Topics

Developers can build advanced themes that add or change CSS classes, map to specific parts of the UI, and change the structure of pages.

[Creating Custom CSS Classes](#) (page 1)

[Customizing Page Structure](#) (page 4)

[Mapping Themes](#)

(page 9) [Best Practices](#) (page 11)

Why Are These Topics "Advanced"?

Well, for one thing, the tasks described here generally require more development work. You'll need to be comfortable working with [FreeMarker](#) templates. Depending on what you do, you might also need to get a more thorough understanding on how the application is structured; editing one FTL file sometimes requires knowing how a few others work, too.

This topic describes how to use the admin console to create, map, and apply themes. There are simpler ways to customize the site's UI if you're looking for simple site-wide changes. You can make basic changes to color schemes and logo through the point-and-click UI described in *Customizing the UI the Basic Way*. You can also download a resource kit, then upload your changes to source files as described in *Customizing UI with the Theme Resource Kit*.

Note: Keep in mind that the changes described here might require more work when its time to upgrade your site. For example, some of the FTL files you can customize depend on libraries or other FTL files that might change with the next version. If you find that the changes you want to make might require changes to the logic or Freemarker code in the FTL files, the Jive Software professional services team can help out.

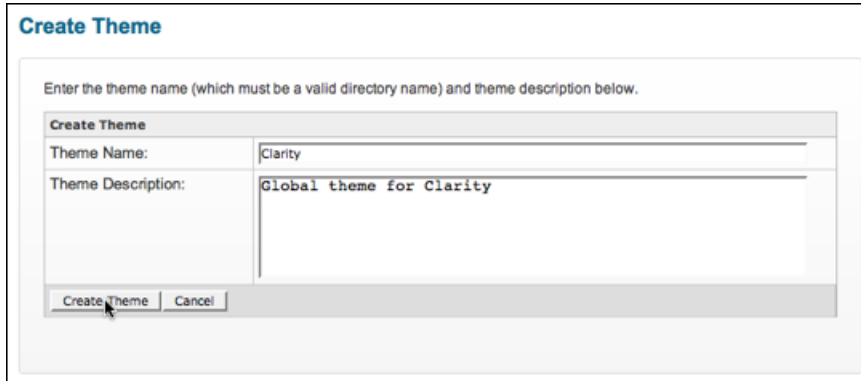
Creating Custom CSS Classes

You customize the CSS of your Clearspace instance by adding or overriding CSS classes in a custom CSS FTL template file. You create a theme, create the custom template, add your CSS classes to the template, then map the theme to the UI. This section walks through the process of making a small change, but larger changes work essentially the same way. Be sure to read [Mapping Themes](#) (page 9) for more information on how to map your themes.

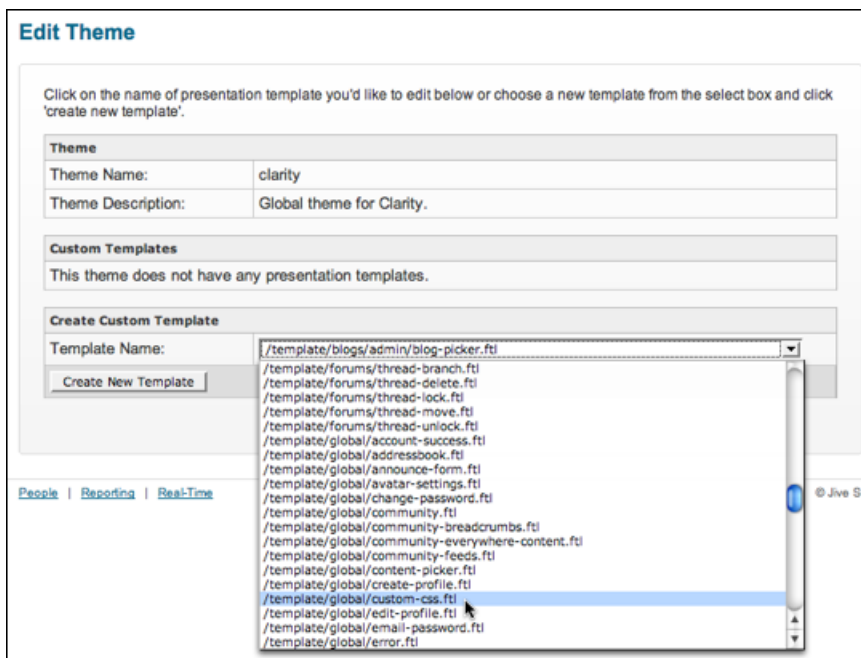
If you create *new* CSS classes, you'll also need to update page-specific FTL files to use the classes you define. See [Customizing Page Structure](#) (page 4) for information.

Note: It's a good idea to plan your customizations ahead of time -- their scope as well as the details of the customizations themselves -- because how you map your theme determines the scope of the changes applied. With those plans in hand, a good way to start is by creating your theme

1. Create a new theme. In the admin console, go to System > Settings > Themes, then click Create New Theme. Give the new theme a name and description, then click Create Theme.



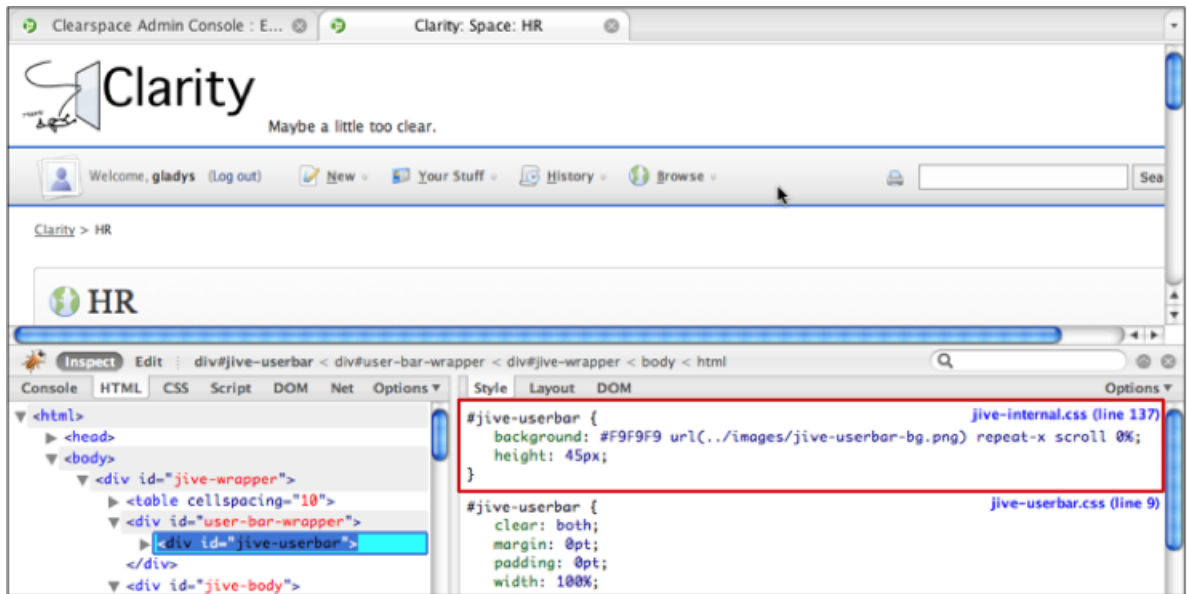
2. Begin editing the theme by clicking its edit icon.
3. Under Create Custom Template, select `/template/global/custom-css.ftl`, then click Create New Template.



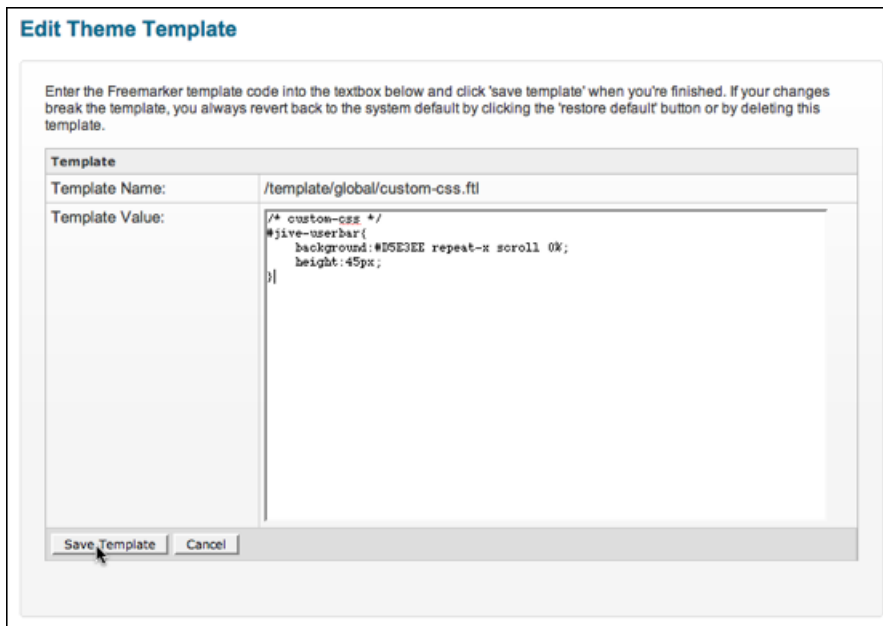
This creates a custom template to which you can add CSS classes. **Note:** When you save your CSS custom template, Clearspace will copy as `<jiveHome>/themes/<theme_name>/global/custom-css.ftl` in your Clearspace installation. You might find that it's easier to simply create and save the template, then open the saved template in your favorite editor.

4. Using a tool such as [Firebug](#) (page 12), figure out the styles you want to add or override. For example, the following illustration shows Firebug focused on the user

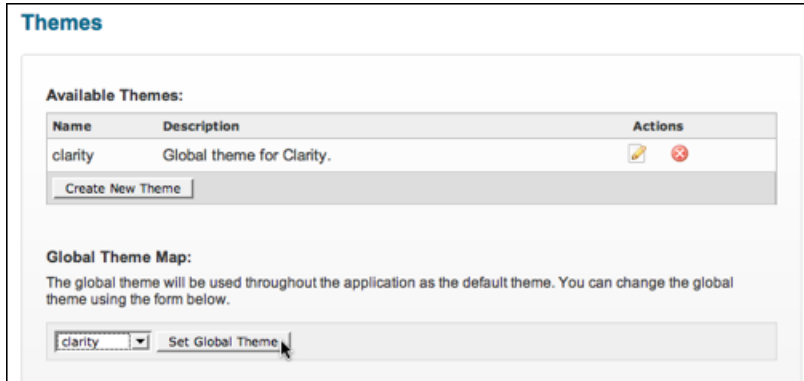
bar (the part with the menus in it), where the jive-userbar CSS class is applied.



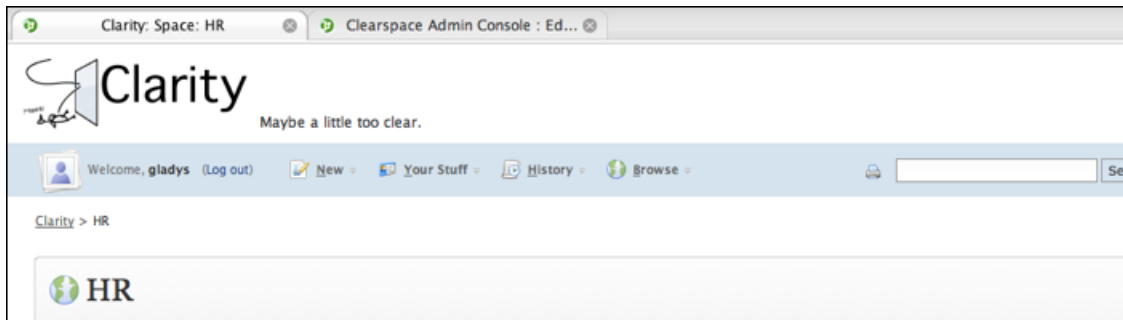
5. To override the style you've identified, simply add a class of the same name to your custom CSS template. (You can even start by copying the classes from the Clearspace instance, then paste them into your custom CSS template.) The following example shows the user bar style updated with a different color and no background image.



6. Save the template when you've added the classes you want.
7. In order to see your changes, you'll need to map the new theme to some aspect of the user interface. For example, if you want the your theme to impact UI everywhere in your Clearspace instance, you'll want to set it as the global theme, as shown below. Note that you can have only one global theme, so plan to add all of your global template changes to a single theme for global mapping.



Here's the user bar with its background color set to blue.



Customizing Page Structure

You change the structure of a Clearspace page by customizing the Freemarker FTL template file that the page is rendered from. An FTL file defines the order of UI elements on a page (or a section of a page), along with the data displayed on the page. The custom template is part of a theme that you map to the user interface.

When adding custom templates to a theme, be sure you're picking a theme that is (or will be) mapped to the part of the UI you want to affect. For more information, see [Mapping Themes](#) (page 9) .

1. Identify the part of the UI that you want to customize, then identify the FTL file that corresponds to it.
2. Create or edit the theme that will contain the custom template.
3. Under Create Custom Template, select the FTL file you're going to customize, then click Create Template.
Note: As with the CSS template, Clearspace will copy your saved template to the `<jiveHome>/themes/<theme_name>` directory of your Clearspace installation. You can edit the FTL file in the admin console, of course, but you'll most likely find it easier to edit in an editor that is designed for code (such as an HTML editor).
4. Make changes to your custom template. The following code takes a blog post title from elsewhere in the page and moves it so that the title follows the blog name.

```

<#include "/template/global/include/form-message.ftl" />

<#if (blog.userCount > 1)>
<span class="jive-groupblog-info">
<#else>
<span class="jive-blog-info">
</#if>
<span class="jive-blog-info-padding">

<h3><a href="{BlogUtils.getBlogURL(blog)}" style="text-decoration:none">${blog.name}</a>:
<a href="{BlogUtils.getPostPermalink(post)}" style="text-decoration:none">${post.subject}</a></h3>

<span class="jive-blog-info-details">

<!-- <strong>${blog.blogPostCount} Posts</strong> -->

<span class="jive-pagination">
<span class="jive-pagination-prevnext">
<#if post.previousPost?exists || post.nextPost?exists >

```

5. If you haven't already, map the theme that contains the custom template to the user interface. This example maps the theme to a URL in order to have it effect all blogs (which aren't contained within a community).

Custom Theme Maps:

Themes can be mapped to various application data such as Spaces and URL patterns. This enables a single Clearspace instance to display multiple skins throughout the application. Theme maps can be applied without a server restart and can be modified anytime.

| Theme | Type | Value | Delete |
|---|------|-------|--------|
| No theme maps are installed at this time. | | | |

Theme:

Map to URL:

Map to Communities:

- Root Space
- R&D
- Products
- HR

You can choose multiple spaces.

Before

Clarity > Blogs > HR Blog > 2007 > April > 23



Vacation policy changes

Posted by [glady](#) Apr 23, 2007 0

I blogged last week on the possibility of changing our holiday schedule. Turns out that quite a few of you thought our old schedule just didn't match up with our seasonal holiday fun (I agree!). But I couldn't help noticing that a lot of the comments I got were about our vacation policy, too. In particular, folks are wondering if they can redistribute that week of vacation we offer at the end of the year.

So this week I thought I'd explain the current policy a bit and offer a suggestion or two. As always, I hope you'll chime in if you have thoughts. (Then again, I know I can count on you for that 😊).

More to come!

Tags: [vacation](#) [time off](#) [hr policies](#)

After



Substituting Phrases in the UI

You can give other names to things people see in the user interface. For example, you can rename "Documents" and "Discussions" to "Articles" and "Conversations." You can make the substitutions in a simple way for the most common terms. There's also a more detailed way to do it.

How It Works

All of your substitutions are made at run time. That is, Clearspace keeps track of what word or phrase you want displayed, then inserts your phrase for the default phrase when someone views a page that includes the phrase. The default phrases themselves from internationalization (i18n) properties files used for localization. Also, your substitutions are associated with a theme. That means that you can make your substitutions globally or by mapping the change to a particular part of the user interface.

Behind the scenes, Clearspace is remembering your substitutions as "substitution rules" that use regular expressions. The rules give you a way to fine tune the substitutions to meet your needs.

Simple Substitutions

In the simple way, you can substitute common phrases that are predefined. In the admin console go to System > Settings > Phrase Substitutions. There you can choose the theme that the substitution should be a part of and a locale in which the substitution should occur. Beneath these selections you'll see a list of the phrases you can make substitutions for. If the phrase you want to change is there, enter your alternatives for singular and plural versions of the phrase.

Click Save to make the substitution (you don't need to restart the application or server for the change to take affect). After you save the rule, you can View Rules to go to the Phrase Substitutions page and see what going on behind the scenes by viewing the details of the substitution rule you're creating.

Phrase Substitutions

Choose a Locale and Theme to which these set of rules should apply.

Locale:

Theme:

Choose alternative phrases.

| | Singular | Plural |
|-------------|--------------------------------------|---------------------------------------|
| Space: | <input type="text"/> | <input type="text"/> |
| Document: | <input type="text" value="Article"/> | <input type="text" value="Articles"/> |
| Discussion: | <input type="text"/> | <input type="text"/> |
| Blog: | <input type="text"/> | <input type="text"/> |

The simple substitution mechanism is designed to account for upper- and lower-case versions of the phrases. However, there are some aspects of the change it doesn't account for. For example, it doesn't account for a change to an article that would precede the word. Substituting "article" for "document" would leave awkward constructions in English, such as "Create a article or upload a file" (where you'd want "Create *an* article or upload a file"). If you're curious about the outcome of the change you're making, click View Rules to see the rules page, where you can get a preview of what the rule is going (you'll need to save your changes first).

Advanced Substitutions

On the Phrase Substitution Rules page, you can change the rule or get more information about it. (You can also get to this page from the Themes page by editing a theme, then clicking Edit Rules.)

Phrase Substitution Rules

Choose a Locale to which these set of rules should apply.

Locale:

| Target | Operator | Pattern | Replace With | Edit | Delete | Add Exception | View Results |
|--------|----------|---------------|--------------|------|--------|---------------|--------------|
| VALUE | REG_EX | \bdocument\b | article | | | | |
| VALUE | REG_EX | \bDocument\b | Article | | | | |
| VALUE | REG_EX | \bdocuments\b | articles | | | | |
| VALUE | REG_EX | \bDocuments\b | Articles | | | | |

On this page you can:

- Edit an existing rule, specifying which value you're substituting and what you're substituting for it.
- Add an exception to a rule. For example, you might want to make a substitution everywhere except where the string containing the phrase contains a certain phrase.
- Delete the substitution rule.
- View the results of your substitution, seeing the context of your changes — what is actually going to be changed. The rule result list shows which i18n keys your change will affect, as well as the "before" and "after" of the change.

| Key | Original Value | New Value |
|---|--|---|
| doc.move.move_document.title | Move Document | Move Article |
| movedoc.description.external | To move this document to another community, use the form below to select the destination community and click "Move Document". | To move this document to another community, use the form below to select the destination community and click "Move Article". |
| prof.watch.doc_filter.listitem | Document | Article |
| sgroup.feeds.crtNewDoc.link | Create a New Document | Create a New Article |
| doc.apprvl.apprvOrReject.text | Approve or Reject This Document : | Approve or Reject This Article : |
| admin.decorator.menu.system.cloud.description | Configure Document Sharing | Configure Article Sharing |
| doc.create.document_title.label | Document Title: | Article Title: |
| doc.create.doc_comments.link | Document Comments | Article Comments |
| doc.ver.edit_document.radio | Edit Document | Edit Article |
| doc.create.title | Create New Document | Create New Article |
| reporting.tags.document | Document Tags | Article Tags |
| admin.decorator.menu.spaces.manage.documents | Document Management | Article Management |
| doc.crtvererr.restrtEdit.button | Restart Document Edit | Restart Article Edit |
| doc.viewer.doc_deleted.text | This document has been deleted. It is only visible to system administrators. If you would like to restore this document to a non-deleted state, you can do so in the Document Management page of the administration console. | This document has been deleted. It is only visible to system administrators. If you would like to restore this document to a non-deleted state, you can do so in the Article Management page of the administration console. |
| reporting.documents.edits | Document Edits | Article Edits |
| doc.move.move_document.button | Move Document | Move Article |

Creating a New Advanced Rule

When you need more fine-grained control over your substitution, you can create an advanced rule. You can do this in the admin console by editing the theme with which the new rule will be associated. In the console, click System > Settings > Themes, then click the edit icon for the theme to which you want to add the rule. On the Edit Theme page, click Edit Rules. On the Phrase Substution Rules page, click Add Rule.

When you're adding a rule this way, you'll have four values to enter.

| Rule Option | Value | Description |
|-------------|--------------|--|
| Target | Value or Key | Whether the rule should be looking for your pattern (what you're replacing) in the i18n values or keys. If you choose "Key" here, the substitution will still be for the key's value, not the key itself. This is handy when you know what a |

| | | |
|--------------|---|---|
| | | particular key is and want to focus on that one in particular. |
| Operator | Equals, Contains, Starts With, Ends With, or Regular Expression | The first four values are self-explanatory. The last specifies that your pattern is a regular expression . |
| Pattern | The original string. | The pattern that should be sought for replacement. For example, if you'd specified "Regular Expression" as your operator, you could enter "\bblog\b" for a pattern that matches the word "blog" (as opposed to "blogger"). The "\b" combination in regular expression can be used to indicate the boundary of a word. |
| Replace With | The replacement string. | The text to substitute instead of the text you enter for a pattern. |

Linking to Image Files

You can include images in your custom pages by linking to them at a directory inside your theme file hierarchy. To keep all your theme resources together, it's a good idea to put your image files in a subdirectory of your theme. So, using the example above, you could create an images directory just inside the theme subdirectory in your jiveHome directory. Here's an example path that contains a mylogo.gif image file:

```
<jive_home>/themes/my_theme/images/mylogo.gif
```

In your FTL file, you can refer to this file using FreeMarker syntax such as the following:

```

```

Mapping Themes

As you plan your customizations, their scope will mean a lot. Do you want the changes to be seen across the UI, only in a particular community, or only in selected patches of the UI reached by certain URLs? The scope decision will be one of the first you make.

You'll use scope to map your themes to the UI so you can test your changes. Without this mapping, it won't be possible to really see how things are shaping up. The scope decisions are built on your design goals. If you're making more than just a few minor changes, you'll find it's useful to think in terms of UI boundaries. Will your changes impact certain kinds of content or certain communities, or certain content *in* certain communities? Your conceptual decision related to UI boundaries translates into implementation decisions about themes and their mappings.

In other words, group your change ideas into batches that correspond to the UI globally, or to communities, or to specific URLs. Then create themes corresponding to those groups, add custom templates to the themes, and map the themes to the UI scope you defined.

Template Hierarchy

Clearspace applies custom templates using the following hierarchy in which more specific levels override broader ones. The precedence from specific to general is URL > Community > Global, as described here:

- If you map a theme to a particular URL, templates in that theme are the only customization used for UI at that URL. All templates from themes mapped to a containing community or from global themes are ignored.
- If you map a theme to a particular community, templates in that theme are used unless you've mapped a URL-specific theme in the scope of that community. Templates in a global theme are ignored.
- If you make the theme the global theme, its templates are used except in any part of the UI where you've mapped a community- or URL-specific theme.

Global

Make a theme the global theme when you want its templates to impact UI across the application. Keep in mind that if you separately customize those same templates in themes that are mapped to communities or URLs, those more specific mappings will take precedence.

Note that the word "global" is use in the path names for some templates. This can be a bit confusing. The name doesn't mean that the templates are always applied globally, but rather than they guide UI that's not specific to certain content types. A "global" template will be applied only to the parts of the UI its containing theme is mapped to, as described above.

Map to a Community

Map a theme to a community for community-specific look and feel. Using custom templates for blogs, communities, or discussions, you can have those content look like they belong together in a particular community. "Global" custom templates will guide the look of their UI elements only in that community.

Keep in mind that if you separately customize those same templates in themes that are mapped to URLs, the more specific mappings will take precedence.

Map to a URL

Mapping a theme to a URL is a great way to customize very specific parts of the UI, such as blogs or user profiles (or even just one person's blog and profile).

When specifying the URL to map to, you use [regular expression](#) syntax. For example, a dot (.) means "any character" and an asterisk (*) means "any number of characters."

For example, blog URLs follow this form:

```
<context_root>/blogs/<blog_name>/<post_year>/<post_month>/<post_date>/<post_title>
```

So the following would map a theme to "any blog posts from 2007":

```
./blogs/./2007.*
```

You can imagine how you might fill a theme with custom templates that guide blog appearance, along with some cross-content UI, then map that theme to this URL to call out content in blog years.

Here are a few more examples in which the Clearspace instance root is merely "clearspace":

All blogs

This reads as "any number of any characters followed by 'clearspace/blogs' followed by any number of any characters."

```
.*clearspace/blogs.*  
  
// Or more universally:  
.*blogs.*
```

Just the blogs main page

This reads as "any number of any characters followed by 'clearspace/blogs' and nothing else."

```
.*clearspace/blogs  
  
// Or more universally:  
.*blogs
```

A specific user's blog

This reads as "any number of any characters followed by 'clearspace/blogs/gladys' and any number of any characters." In other words, this maps to all posts in Gladys' blog.

```
.*clearspace/blogs/gladys.*  
  
// Or more universally:  
.*blogs/gladys.*
```

Best Practices

As you make your changes to the UI, try the following to make your work easy and straightforward.

Make Your Plans

Think about your changes in terms of the UI regions they'll touch, then plan your themes as groups of templates corresponding to those UI regions. For more information, see [Mapping Themes](#) (page 9) .

Also, make a list of the FTL files you'll want to customize for each theme. Note that this could be different custom versions of the same file for different themes.

Use the Right Tools

Use browser-based tools such as the *Firefox Web Developer* and [Firebug](#) plugins for the Firefox web browser. These tools are useful for figuring out which CSS classes you should edit to change a particular part of the UI. Here's a shot of Firebug displaying HTML markup and CSS style corresponding to a blog post title.



Use the admin console to create and save your templates, but edit them with your own editor (preferably one that features syntax coloring). The Clearspace admin console provides a way to edit these files, but it's pretty basic. After you save templates, you'll find them at a path like the following:

```
<jive_home>/themes/<theme_name>/<template_path>
```

For example, if you create a custom template based on custom-css.ftl and associate it with a my_theme theme, you'll find it here:

```
<jive_home>/themes/my_theme/template/global/custom-css.ftl
```